2005

Information System Development Methodologies as Learning Systems

Anand Jeyaraj
University of Missouri - St Louis, Jeyaraj@umsl.edu

Vicki L. Sauter
University of Missouri - St Louis, vicki.sauter@umsl.edu

Information System Development Methodologies as Learning Systems

Anand Jeyaraj
University of Missouri – St. Louis
jeyaraj@umsl.edu

Vicki L. Sauter
University of Missouri – St. Louis
vicki.sauter@umsl.edu

ABSTRACT

Although information system development methodologies supposedly improve development processes and end-products, information systems have continued to fail. In practice, methodologies have not been universally accepted, and even when accepted, not consistently used. Often development teams adapt methodologies to respond to perceived problems in past applications and/or specifics of the project under consideration. While it is possible that a combination of these factors contributed to less-than-effective system development processes or final end-products, we believe that there may be a subtler explanation for such failures. Although methodologies recommend best practices for system development, they rarely prescribe mechanisms to capture or evaluate problems encountered during system development. Specifically, methodologies do not include "learning" mechanisms that may be useful in improving the effectiveness of processes and end-products. We outline a meta-methodology for system development methodologies, which describes ways in which learning at different levels can be used to improve the effectiveness of system development methodologies.

Keywords

Information System Development, Methodology, Learning.

INTRODUCTION

Throughout the history of system development projects, researchers and practitioners have reported that a vast majority of information systems (IS) fail. Over thirty years ago, Naur and Randall (1968) brought attention to the "software crisis" associated with system failure. The GAO (1979) noted that less than 2% of the software contracts could be used efficiently, and only another 3% could be used when major changes were made to them. Standish (1995) reported that about 30% of the US projects that year were cancelled, and over 50% were completed using more time and/or money than had been budgeted, often with reduced functionality from the original plan. Even today (Dalcher and Drevin, 2003), researchers report significant software project failure rates worldwide. Some estimate that 85% of all systems fail either because they are not finished, do not fulfill users' needs, take longer and/or cost more than estimated.

System failure was traditionally associated to ad hoc development procedures (Wasserman, 1980; Avison and Fitzgerald, 2003). In response, consultants and academics systematized successful development procedures into information system development (ISD) methodologies. Over one thousand methodologies have been proposed over the years, including Systems Development Lifecycle, Information Engineering, Prototyping, Rapid Application Development, and Object Oriented Analysis and Design (Sutcliffe, 1991; Jayaratna, 1994). Each methodology prescribed "best practices" for ISD procedures and policies that supposedly improved the system development process as well as the final IS end-products. An experiment by Howard et al. (1999) reinforced the benefits of such systematic approaches when it revealed that methodology-assisted groups produced significantly better systems than groups that did not use any methodologies. Even with this information, and greater reliance on methodologies, system failures still plague the industry.

Empirical evidence suggests that methodologies have not been adopted with much rigor or vigor. Even with the evidence of a benefit, not all ISD teams use formalized methodologies. Fitzgerald (1998) reported that 60% of the 162 organizations responding to a survey indicated that they were not using methodologies. Further, there is not adequate guidance for which methodology to adopt; a given methodology may or may not be a good solution for a particular project. Saarinen (1990) reported that the recommendations for situational choice of methodologies were not completely adopted in practice. Moreover, even when formalized methodologies were used for system development, not all procedures and recommendations of the methodologies were adopted. Fitzgerald (1998) reported that only 6% of the 162 organizations rigorously followed a commercial methodology. Finally, the recommendations of methodologies were often customized for the specific contexts. For example, Jenkins, Naumann, and Wetherbe (1984) reported that a large percentage of those using a methodology

modified it for their projects. While it is possible that a combination of these factors contributed to less-than-effective system development processes or final end-products, we believe that there may be a subtler explanation for such failures.

Although methodologies recommend best practices for system development, they rarely prescribe mechanisms by which problems experienced during system development can be recorded and evaluated. In essence, methodologies do not include "learning" mechanisms that prescribe *systematic* procedures by which methodology usage may be monitored and evaluated. We argue that methodologies should necessarily incorporate learning such that system development processes and end-products are improved. We present a meta-methodology, namely Universal Methodology for Systems Lifecycle (UMSL), which incorporates feedback systems to implement learning within methodologies. UMSL provides recommendations on how ISD methodologies can function as learning systems. It also describes how feedback at various levels, such as system development teams, adopter organizations, and standards organizations, are beneficial, and can be integrated for improving the effectiveness of processes and end-products.

METHODOLOGIES AS LEARNING SYSTEMS

> **Hypothetical ISD Scenario**: Consider the final IS product that resulted from a system development project. Further consider that some activities executed by the new system run counter to the expectations of a particular stakeholder group. Different interpretations of the scenario may highlight different reasons for the discrepancy. However, assuming that the system developers did a faithful job of translating requirements to the final system, the problem may have been faulty requirements definition or insufficient requirements gathering.

Learning entails changes in processes, structures, and assumptions that underlie the operations of an organization. Different levels of learning, such as zero, single-, double-, and triple-loop learning have been proposed to differentiate the intensity of change in behavior (Argyris and Schon, 1978; Snell and Chak, 1998). Zero learning represents the failure to receive feedback on actions.

Single-loop learning involves "adjusting action to achieve the desired outcome" (Argyris and Schon, 1978; Snell and Chak, 1998). Single-loop learning involves adaptive changes, improves the efficiency and effectiveness in the short-term, and leaves the existing values and norms unchanged (Korth, 2000). In the hypothetical above, the system developer would attempt to rectify the situation by clarifying the expectations of the stakeholder group and then implementing the changes on the final IS product. Thus, the system developer would only fix the discrepancy between the expected and actual outcomes, but would not be concerned about the underlying reasons for the discrepancy.

Double-loop learning involves "transforming mental models to generate new meanings and activities" (Argyris and Schon, 1978; Snell and Chak, 1998). Double-loop learning involves fundamental changes, takes a long-term perspective, and questions the underlying values and norms (Korth, 2000). In the hypothetical above, in addition to fixing the discrepancy, the system developer would also examine the processes that guided system development. Thus, the system developer may actually discover that the requirements gathering process probably did not directly involve the stakeholder group. It is also possible that the system development process would be altered to include a more inclusive role for stakeholder group participation in requirements specification.

Triple-loop learning involves "inventing new processes for generating mental maps" (Snell and Chak, 1998). Triple-loop learning involves the reexamination of existing practices and the invention of new practices. In the hypothetical above, the system developer may conduct an in-depth evaluation of the entire system development process, and may actually determine that information systems may be developed using completely different processes that are more efficient and effective.

META-METHODOLOGY: UNIVERSAL METHODOLOGY FOR SYSTEMS LIFECYCLE

The Universal Methodology for Systems Lifecycle is an archetype for system development methodologies and incorporates single-, double-, and triple-loop learning mechanisms. The fundamental process underlying the UMSL system is the translation of functional requirements into an IT artifact (Figure 1). The system development team is generally responsible for this process. These processes are enacted within a "design context," i.e., the project context in which the development of the IT artifact is accomplished.

The process assumes that the functional requirements have been gathered and are available for verification. The expectation is that the "right" IT artifacts conforming to the functional requirements would be built. Feedback is essential to evaluate if the right IT artifacts are obtained through this fundamental process; F1 in Figure 1 represents that feedback within the design context. For instance, the system development team may tweak the IT artifact to be in alignment with stated requirements if the actual outcomes are different from the expected outcomes. The system development team is more likely to be interested

in building the "right" IT artifacts rather than adopting the "right" process. Thus, single-loop learning is the norm within the design context.

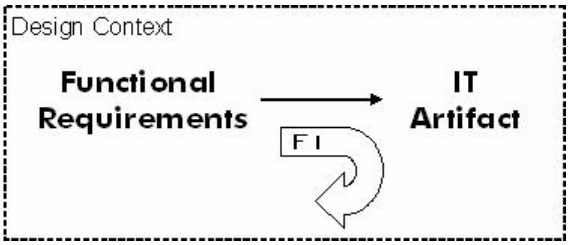

Figure 1. Design Context

Single-loop learning may not be the most fruitful approach to learning since it has some important limitations. First, the system development team would not reflect on the reasons why the IT artifact did not meet expectations. Or, if they did reflect on the experience they might draw erroneous conclusions. Second, left unquestioned, subsequent development efforts may also witness the same issues with the artifact, and need to address the same problems. Hence, the development effort would be inefficient, or if the team is not as skillful, ineffective.

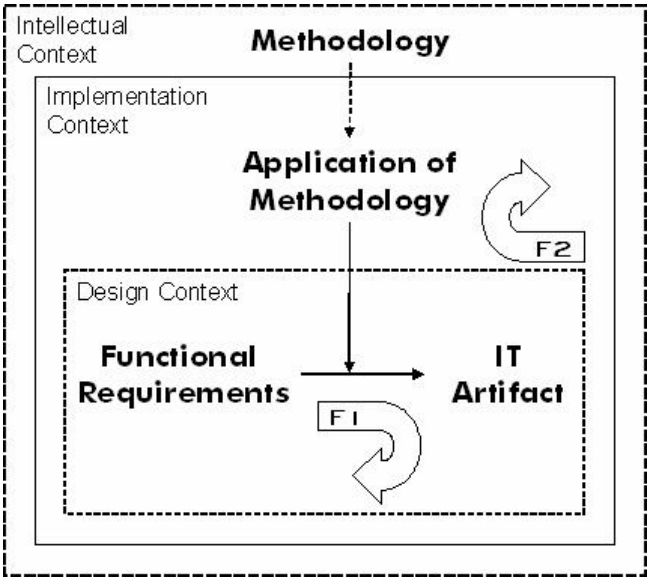

Figure 2. Implementation and Intellectual Contexts

The effectiveness of the fundamental system development process and the final IS artifact may actually be enhanced by the application of a methodology (Figure 2). This is not surprising since methodologies prescribe systematic ways in which the IT artifact may be developed and hence the chances of building the "right" artifact are supposedly increased. However, the application of a methodology is performed within an "implementation context," i.e., the site of the system development effort. The unique characteristics of an implementation context may also dictate the ways in which the methodology is applied to the system development effort. The application of a methodology is also guided by the particular methodology selected for development (Figure 2). The methodology chosen for development might include such phases as requirements gathering, requirements definition, design, testing, and implementation. The particular ways in which the specific activities, such as requirements gathering, are done is to a large extent guided by the implementation context. Thus, the implementation

context is set within a broader "intellectual context," i.e., the particular methodology typically prescribed by a standards organization.

F2 in Figure 2 represents the feedback activities within the implementation context. The practices adopted by the system development team and the effectiveness of those practices may be evaluated across team efforts to determine the problem areas in the application of a methodology. It may be determined, for instance, that the IT artifact repeatedly failed to meet expectations of stakeholder groups. These insights may raise questions about the particular processes, such as requirements gathering, that were followed by the system development teams. This might, in turn, lead to customization of those processes for the specific implementation context regardless of the recommendations of the intellectual context. Thus, double-loop learning is the norm within the implementation context.

Double-loop learning offers certain advantages over single-loop learning. First, the likelihood of building the "right" IT artifacts is increased due to the critical evaluation of the application of methodology within the implementation context. The evaluation paves the way for the system development team to adopt the "right" practices in dealing with the fundamental system development process. Second, double-loop learning enables organizations (implementation contexts) to evaluate their experiences with methodologies *systematically* and adapt their changes in the methodology.

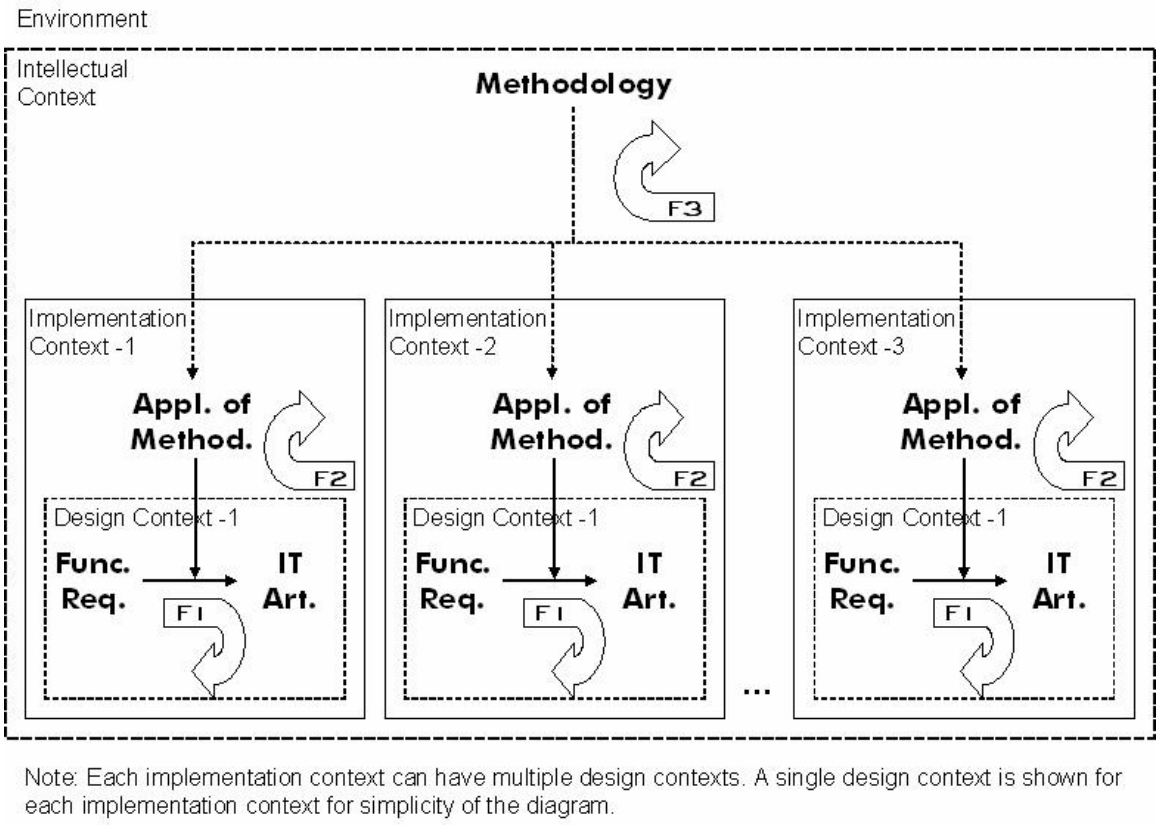

Figure 3. Universal Methodology for Systems Lifecycle

A methodology may be adopted by multiple organizations with different needs and peculiarities. Figure 3 illustrates how a single intellectual context supports multiple implementation contexts. The collective of intellectual contexts represents the acceptance of a particular methodology for practice. The development and sharing of the methodology is dependent on the technological advances and capabilities that prevailed when the standards were created. The adaptation of that methodology is generally affected by its "environment," i.e., the broad institutional factors such as the technological, political, professional, cultural, and ethical standards.

F3 in Figure 3 refers to the feedback activities within the intellectual context. The intellectual context provides the standards that guide the implementation contexts to adopt ideal practices for system development. It would be ideal to evaluate the effectiveness of the intellectual context across different implementation contexts by examining the application of the methodology in various contexts. Said differently, we can improve the methodology only when we can *systematically* evaluate its effectiveness and efficiency after team-specific, project-specific, organization-specific, and even industry-specific issues have been controlled. For instance, if a particular best practice recommended by the intellectual context (methodology) has failed in a sufficiently large number of implementation contexts (companies) or design contexts (projects), then it may make sense to reevaluate the processes and determine necessary modifications to the methodology. Thus, triple-loop learning is the norm within the intellectual context.

CONCLUSION

This is a research-in-progress paper. We are developing the processes and metrics associated with the feedback loops at various levels. We will present in more detail at the conference.

REFERENCES

1. Argyris, C., and Schon, D. A. (1978) *Organizational Learning: A Theory of Action Perspective*, Addison-Wesley, Reading, MA.
2. Avison, D. E. and Fitzgerald, G. (2003) Where Now for Development Methodologies? *Communications of the ACM*, 46, 79-82.
3. Dalcher, D., and Drevin, L. (2003) In *2003 Annual Research Conference of the South African Institute of Computer Scientists and Information Technologists on Enablement Through Technology*, South African Institute for Computer Scientists and Information Technologists.
4. Fitzgerald, B. (1998) An Empirical Investigation into the Adoption of Systems Development Methodologies, Information & Management, 34, 317-328 *Information & Management*, 34, 317-328.
5. General Accounting Office. (1979) Contracting for Computer Software Development-Serious Problems Require management Attention to Avoid Wasting Additional Millions, Washington, D.C., USA: General Accounting Office Report to the Congress by the Controller General of the United States, FGMSD 80-4, November 9.
6. Howard, G. S., Bodnovich, T., Janicki, T., Liegle, J., Klein, S., Albert, P. and Cannon, D. (1999) The Efficacy of Matching Information Systems Development Methodologies with Application Characteristics - An Empirical Study *Journal of Systems and Software*, 45, 177-195.
7. Jayaratna, N. (1994) *Understanding and Evaluating Methodologies, NISAD: A Systematic Framework*, McGraw-Hill, Maidenhead, UK.
8. Jenkins, A., Naumann, J. and Wetherbe, J. (1984) Empirical Investigation of Systems Development Practices and Results *Information & Management*, 7, 73-82.
9. Korth, S. J. (2000) Single and Double Loop Learning: Exploring Potential Influence of Cognitive Style *Organization Development Journal*, 18, 87-98.
10. Naur, P., and Randall, B. (1968) In *NATO Conference: Scientific Affairs Division*, Brussels: NATO.
11. Saarinen, T. (1990) Systems Development Methodology and Project Success: An Assessment of Situational Approaches *Information & Management*, 19, 183-193.
12. Snell, R., and Chak, A. M. (1998) The Learning Organization: Learning and Empowerment for Whom? *Management Learning*, 29, 337-364.
13. Standish Group. (1995) Chaos 1995, Dennis: MA.
14. Sutcliffe, A. G. (1991) Object-Oriented Systems Development: Survey of Structured Methods *Information and Software Technology*, 33, 433-442.
15. Wasserman, A. I. (1980) Information System Design Methodology *Journal of the American Society for Information Science*, 31, 5-24.